A Modified Edge Removal Stiener Tree Heuristic for Global Routing in VLSI Design

Apichat Terapasirdsin[1], **William F. Rotach**[2], **Naruemon Wattanapongsakorn**[3], and **Patrick H. Madden**[4], Non-members

ABSTRACT

Routing is a key stage for VLSI physical design. Steiner tree construction is a well studied topic in design automation. There have been a number of significant theoretical advances in the past few years. The focus of this paper is on combining the speed and solution quality of a high quality Steiner heuristic with the reality of modern routing. Practical designs contain routing congestion and blockages; routing is implemented across multiple layers. Each routing layer has preferred directions, and connecting vias have significant cost. In a modern design, many trees are in competition with each other for scarce routing resources. The objective is not to simply build trees with the lowest length; they must also be low cost. We present an approach that is as fast as spanning tree construction, while accurately modeling routing costs.

Our work extends an earlier Steiner tree heuristic algorithm by adding the ability to minimize the routing congestion without altering the computational complexity of the underlying algorithm. We compare our CAST algorithm with the capacitated minimum spanning tree (CMST) and the ER algorithm finding that our approach offers impressive reduction in congestion cost in average about 23.1% and 63.4%, respectively.

Keywords: Steiner tree, Congestion, Global Routing, Rectilinear Distance

1. INTRODUCTION

Interconnect topology is a well studied area in integrated circuit design. As transistors have become faster, the interconnect wires have taken an increasing share of system delay [1]. The wiring is now a limiting factor in the performance of advanced designs. Interconnect capacitance accounts for a great deal of switching power, and detours insertion to avoid routing congestion can introduce additional delay. Extreme cases of routing congestion can even prevent feasible physical design. As a result, additional space must be inserted to obtain a correct solution, causing larger size of the circuit design.

Much of the study of interconnect has focused on relatively simple metrics. "Manhattan distance," in particular, has received a great deal of attention. While the metric is a reasonable approximation of interconnect cost, it fails to capture a number of important elements. Circuit interconnect almost always has "preferred direction," with some layer alternating directions [2]. Connecting the layers together are obtained with vias, which have high cost, and comparatively giving high failure rates; these are not modeled at all in traditional planar formulations. Further, the Manhattan distance metric does not capture routing congestion. So, when constructing an interconnect topology, one may wish to choose tree edges to avoid congested areas.

In this paper, we adapt the well known edge-removal Steiner tree heuristic [3] to handle routing congestion by using an efficient heuristic routing cost caching approach, so that we maintain low run times, while accurately capturing congestion cost. We demonstrate our approach by applying the heuristic approach to various selected nets on ibm01 (ISPD98 IBM Benchmark).

The remainder of this paper is organized as follows. First, we briefly describe routing metrics, and some of the leading methods for generating interconnect topologies. We then describe routing congestion, and provide a simple but effective way to determine the routing cost between any two arbitrary points. Following this, we then describe the integration of routing cost calculations with the edge-removal heuristic, and then present experimental results. We conclude the paper with a summary of our research findings.

2. PREVIOUS WORK

There is an abundance of prior work on the Steiner problem. We will briefly touch on some relevant works as follows.

Manuscript received on December 27, 2008 ; revised on May 7, 2009.

[1,3] The authors are with The Department of Computer Engineering, King Mongkut's University of Technology Thonburi 126 Pracha-Uthid Road, Bangmod, Tungkru, Bangkok, 10140, Thailand, Tel: +66(08)-37975555 , E-mail: apichatter@yahoo.com[1], s5510111@st.kmutt.ac.th[1], naruemon@cpe.kmutt.ac.th[3]

[2,4] The authors are with The Department of Computer Science, State University of New York at Binghamton, New York, USA., E-mail:

2.1 Manhattan Distance and the Steiner Problem

Manhattan (or "rectilinear") distance minimization is the most common objective when generating interconnect topologies for circuit design. It is easy to compute, and roughly captures the spirit of the problem. Chiang [4] proposed an algorithm for obtaining a weighted (rectilinear) Steiner tree in the plane. The proposed global router at each step finds a weighted Steiner tree for a net, where weight of a region represents its "complexity". This Weight of the region is dynamically changing.

Generating low cost spanning trees is trivial. An Example of well known algorithms is presented by Prim [5] and Kruskal [6]. The interconnect problem becomes interesting, when we include the possibility of introducing Steiner points. Hanan [7] showed that for v vertices, there were $O(v^2)$ possible Steiner points for the Manhattan distance metric. While the added freedom of additional points allowing for tree length reductions is considered (of about 11% on random graphs), The problem becomes intractable.

Using computational geometry techniques, Kahng and Robins [8] developed an effective approach that iteratively inserted a single Steiner point at a time, seeking the best improvement at each step. By limiting the number of Steiner points that must be considered, and with efficient data structures, the approach produced excellent results with reasonable run times. There were further advances, with the GeoSteiner [9] approach from Warme, Winter, and Zachariasen, combining computational geometry and linear programming to obtain optimal solutions for surprisingly large graphs.

More recently, Ozdal [10] proposed an iterative rip-up and reroute (RNR) global routing algorithm that guides the routing iterations out of local optima through effective usage of congestion histories. Chu presented "FLUTE [11]," a surprisingly fast Steiner algorithm that is able to find optimal solutions for relatively small problems through table lookup approach.

2.2 Multiple Layers

Missing from most Steiner tree approaches are factors that impact integrated circuit design. First, it should be obvious that most interconnect is not planar, i.e., circuits are fabricated with multiple layers of interconnect metals, normally in alternating directions. A large number of vias are used to perform inter-connections of metals as well as layers, causing relatively high electrical resistance, and consuming a great deal of routing resources.

Fig. 1 presents a routing problem in circuit design. In general, when connecting a set of points, it is typically desirable to utilize minimum length interconnect trees. In the circuit routing, however, some regions may be heavily congested, and it may be worthwhile

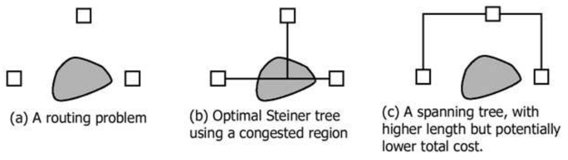
(a) A routing problem
(b) Optimal Steiner tree using a congested region
(c) A spanning tree, with higher length but potentially lower total cost.

Fig.1: *Connecting a set of points*

to trade slightly longer wiring length with lower total cost obtained from less congested region due to lower peak routing demand. The cost of routing in a congested region in the circuit determines the suitable topology of the routing. Geometric techniques can greatly simplify the Steiner problem, but missing to take care of this fundamental issue. Further, each layer may have a unique routing width, giving a distinct cost added to the total routing cost. The Steiner heuristic that forms the basis for our work is adapted to handle differing routing costs on multiple circuit layers [2].

2.3 Congestion in Global Routing

With fixed-die designs, routing congestion has become a significant concern [12]. The routing resources are limited, and if too many connections attempt to use one portion of the routing area, significant detours are unavoidable. In rip-up and reroute based global routers, it's common to assign resource cost based on utilization [13]. Thus, the "cost" of a connection is not from the physical distance, but the sum of the costs of individual routing tiles (usually determined by Dijkstra's algorithm [14]). In this paper, we use a simulated annealing algorithm (SA) [28] to select the direction of routing path obtained from the Dijkstra's algorithm.

The focus of this paper is on quickly finding high quality Steiner trees, under a congestion based cost metric. The problem we address is topology design using global routing based costs, as shown in figure 1(c). In this figure, we face with a routing problem containing three points While this would be trivial to solve with a simple distance metric, a congested region (with higher cost) may make a spanning tree configuration preferable. There are several effective graph-based Steiner heuristics (such as one by Minoux [15]), but these are designed with no underlying structure. Our research focus on integrated circuit routing, and we present an approach that is as fast as a spanning tree construction, while accurately modeling routing costs.

2.4 Simulated annealing

In this paper, we apply an edge removal heuristic by using a simulated annealing algorithm (SA) [28], where an initial solution is repeatedly improved by making small alterations until further improvements cannot be made by such alterations. Unlike greedytype local search algorithms, the SA algorithm can avoid entrapment in a local minimum by allowing occasional uphill moves which deteriorate the objective function value. The uphill move is allowed with the probability given by $exp\ (-\Delta/T)$, where T is a control parameter called the temperature, and Δ is the difference between objective function values of the current and neighborhood solutions. The temperature is initially set with a certain method and gradually lowered in a predetermined method, called the cooling schedule. The following shows how the SA algorithm is implemented for the global routing problem. The temperature T is initially set high. Therefore, the probability of accepting a move that increases the objective function is initially high. The temperature is gradually decreased as the search progresses. That is, the system is cooled slowly. At the end, the probability of accepting a move that decreases the objective function value becomes vanishingly small. In general, the temperature is lowered in accordance with an annealing schedule. The most commonly used annealing schedule is called exponential cooling, which begins at some initial temperature, T_0, and decreases the temperature in steps according to $T_{i+1} = \alpha T_i$ where $0 < \alpha < 1$. Typically, a fixed number of moves must be accepted at each temperature before proceeding to the next. The algorithm terminates either when the temperature reaches some final value, T_{final}, or when some other stopping criterion has been met.

3. CONGESTION AWARE STIENER TREE (CAST)

One may anticipate that when considering routing congestion in an interconnect tree, the approach must necessarily rely on shortest path algorithms (and consequently, there would be a dramatic increase in run time). In practice, however, this is not the case.

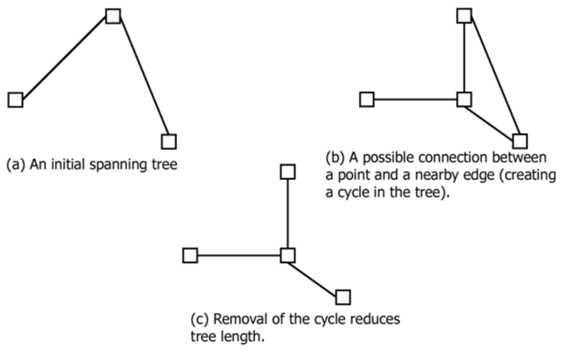
(a) An initial spanning tree
(b) A possible connection between a point and a nearby edge (creating a cycle in the tree).
(c) Removal of the cycle reduces tree length.

Fig.2: *The edge removal heuristic.*

While working on congestion prediction and estimation, Westra, Bartels, and Groeneveld [16] noted that almost all point-to-point connections generated by a commercial global routing tool contained a single bend. While the optimal path in a graph might be expected to meander, this does not seem to occur frequently. One might expect that this is an artifact of considering via cost as part of the routing objective: if vias are expensive, changes in direction are penalized.

From Fig. 2, given an initial spanning tree, potential connection points between edges and vertices are determined. If a connection between an edge and a vertex is made, it will create a cycle in the tree; the longest (or highest cost) edge on the cycle can then be removed. The global routing problem can be modeled as a grid graph $G(V, E)$, where each vertex v_i represents a rectangular region or module of the chip, so called a global routing cell (G-cell), and an edge e_{ij} represents the boundary between v_i and v_j with a given maximum routing resource m_{ij}. A global routing is to find paths that connect the edges, E, inside the G-cells through $G(V, E)$ for every net, (v_i, v_j) [19].

```
Algorithm CAST L-Shape global routing
int layer1, layer2
// connect x1, y1, z1, to x2, y2, z2

Cost1 = Horizontal_routing first and Vertical_routing second;
Cost2 = Vertical_routing first and Horizontal_routing second;
Minimum Cost = min(Cost1, Cost2);
Routing with Minimum Cost is chosen;

// Horizontal_routing

for (layer1 = 0; layer1 < max_layer; ++layer1)
  for (layer2 = 0; layer2 < max_layer; ++layer2)
   {
    cost = cost from (x1, y1) to (x2, y1) on layer 1
      plus  cost from (x2, y1) to (x2, y2) on layer2
      plus via cost from z1 to layer 1
      plus via cost from layer1 to layer 2
      plus via cost from layer 2 to z2
   }
//Vertical_routing
// Similar technique from Horizontal routing is applied for vertical routing (Y direction)
```

Fig.3: *Algorithm CAST L-Shape global routing.*

From Fig. 3, the CAST L-Shape global routing algorithm illustrates global routing in 3-dimension. To understand easily, we demonstrate it in x-y axis, where the MST (Minimum Spanning Tree) has been obtained. With the constraint that point-to-point connections contain at most one bend, we can adapt the edge removal heuristic algorithm to optimize the routing congestion, while maintaining the excellent run times. The edge removal heuristic algorithm of Borah, Owens, and Irwin [3] is chosen. The detail of the algorithm is described in the next section.

3.1 The Edge Removal Heuristic Algorithm

The edge removal (ER) heuristic algorithm that forms the basis of our work is remarkably flexible, as well as easy to implement. The heuristic algorithm begins with a minimum spanning tree, and then attempts to improve itself through a series of passes.

In each pass, we consider introducing a connection between any two vertices with each edge in the graph. This connection would create a Steiner point (the cheapest connection for the vertices and edges), and also a cycle. Redundant edge is removed from the graph in order to minimize the connection cost. The potential connections and gains can be computed quickly with $O(V)$ using depth-first search algorithm. After determining possible modifications that can reduce tree cost, they are applied in order of benefit; it is easy to determine if one modification precludes another. Each pass is at most $O(V^2)$, where the depth-first search algorithm is $O(V+E)$, and the number of edges, E, in the spanning tree is $O(V)$. Typically, the iterative process fails to find further improvements after about three passes. An edge-removal operation is illustrated in Fig. 2.

3.2 Routability and Wiring Length

Routability can be estimated by the number of over-flows which indicates that routing demand locally exceeds the available routing capacity [25, 26]. The number of overflow between any two adjacent units is one.

• Wiring Length is an important metric for placement as well as routing. However, it is less concern for global routing, as routing all wires with shortest path algorithms will result in minimum or near-minimum wiring length [24]. However, there can be huge difference between solutions of the same wiring length in terms of routability.

• Runtime is fairly significant, as global routing is significantly related to placement and detailed routing. Global routing is performed first, followed by detailed routing. Parasitic information from the lower level design is may be needed to feedback to the higher level of design flow for the design convergence. With efficient global routing algorithm, the runtime for the circuit design can be minimized.

3.3 Adding Congestion to the Heuristic

When considering routing congestion, the observation that connections are typically L-shaped [16] becomes very useful. Rather than computing connection cost using Dijkstra's algorithm, we can instead simply sum up the costs of routing tiles of the two L-shaped paths for a pair of points.

In rip-up and reroute approaches, routing cost is a function of the demand for graph edges. In many recent works, cost increases abruptly when demand reaches (or approaches) its capacity. In the CAST routing tool, routing cost is a linear function of demand, so that we can pre-compute the costs of series of tiles. Rather than adding costs in a step-by-step way, we can compute costs in a constant time. This method is illustrated in Fig. 4.

Pre-computing routing costs can be time consuming. However, in the case of a global routing, this can be done once per pass for the router. In [17], Steiner trees were repeatedly rebuilt, to balance the routing utilization of multiple layers. Thus, with each pass, overall routing demand changed slowly. As a result, we can anticipate only moderate changes in congestion (and routing cost) with each pass of tree construction, and there is little need to rapidly update the cost surface.

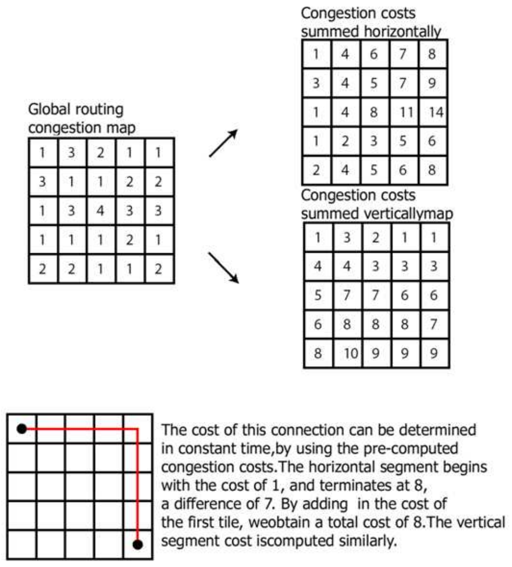
Global routing congestion map
Congestion costs summed horizontally
Congestion costs summed verticallymap
The cost of this connection can be determined in constant time,by using the pre-computed congestion costs.The horizontal segment begins with the cost of 1, and terminates at 8, a difference of 7. By adding in the cost of the first tile, weobtain a total cost of 8.The vertical segment cost iscomputed similarly.

Fig.4: *Congestion map utilization for global routing.*

From Fig. 4, congestion map utilization for global routing is shown in the upper left portion of the figure. The routing cost is based on resource demand, rather than the routing distance alone. As most connections each contains a single bend, we can pre-compute horizontal and vertical costs, in order to determine the cost of any point-to-point connection in a constant time.

3.4 Global Routing with Congestion Estimation

Our routing approach extends previous work through the introduction of a congestion estimation method. This influences the routing of individual connections, improving the solution quality. In this section, we first describe the routing method of [11] in more detail. We then present our method of congestion prediction. The routing costs are influenced by the congestion estimates; our objective is to disperse routes from areas that we expect to have routing congestion.

$$\begin{array}{ll} y=m_1x+c_1, & 0<=x<=k_1 \\ y=m_2x+c_2, & k_1<x<=k_2 \\ y=m_3x+c_3, & k_2<x<=k_3 \end{array}$$

Fig.5: *The CAST routing cost.*

We consider the routing cost to be slightly increasing at the beginning where the demand is still low until it gets to a certain limit where more routing demand greatly affect to the routing cost; the routing can be obtained with sacrificed cost up to a certain capacity. Beyond this limit, more routing can still be obtained with a different cost rate. In this way, we pay more attention on the routing demand that is getting close to the routing capacity.

Fig. 5 shows the CAST routing cost function. We assign a unit cost to an edge initially and the cost is slightly increase linearly until it reaches 20% of the routing capacity, where the cost starts to rapidly increase linearly until reaching a maximum cost. The parameters m_1, m_2, m_3 are routing demand scales , x is number of nets$(n), k_1, k_2, k_3$ are edge capacities, and c_1, c_2, c_3 are congestion parameters. k_1 is the initial step. In this experiment we set m_1= 0.2, c_1= 0.8, m_2= 6, c_2= -22, m_3= 0.2, and c_3= 7. If the steps keep repeating on the same route, the cost will increase slowly until it reaches k_2, where the cost increases dramatically at this stage and beyond. The routing cost behavior in Fig. 6 shows number of repeating routes(R) that the cost keeps increasing slowly from number of repeating routes 1-4, then at the 5th repeated route, the cost suddenly increases dramatically to force the process to choose a different route. Then, if the congestion solution is found after a few steps at some other routes, the process can return to this route again while the cost keeps increasing slowly. This method is a modified version of the classical cost function (step function)

As is done in most global routing tools, we monitor the number of routes in our routing graph, and use this to adjust routing cost. We consider this to be an extremely important issue in the construction of an effective global routing tool. In [20], it was observed that having routing cost increase linearly with congestion was far more effective than having a "step" cost function; these observations were confirmed in [27]. Despite this, several recent global routers such as Labyrinth[22] and Albrecht[23] have their routing costs increasing abruptly when the number of routes on a graph edge reaches the edge capacity. Our routing cost function is illustrated in Fig. 6, where the maximum routing cost is set to 10.

The routing costs are influenced by the congestion estimates; our objective is to disperse routes from areas we expect to be congested. Prior to initial routing, the method of Linsker [22] has no indication as to where congestion is to be expected. The first routes may pass through areas that will have high demand, even when there are alternatives. During rip-up and reroute, the routings of some wires impacts the routes considered for others. Our objective with congestion estimation is to minimize the number of poorly routed connections in the early stages of routing, and to provide an effective method when there are multiple routes with similar cost. In [21], a similar technique is considered, but dynamically updated routing cost because no capacity constraints

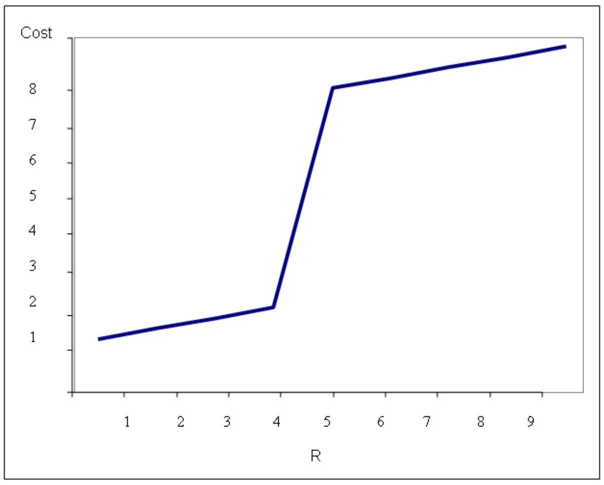

Fig.6: *CAST routing: a piecewise linear function of demand.*

were included with their benchmarks. The intuition for the "linear" cost function is clear: early routes will encounter an uncongested routing graph, and will utilize the resources without consideration for the demands of later routes. Later routes will then detour around congested regions, increasing the overall routing demand considerably. During rip-up and reroute, the connections routed in the early stages will have difficulty in being rerouted, because of the added wire length.

We demonstrate our approach by applying the heuristic algorithm to randomly select nets on ibm01 (ISPD98 IBM Benchmark, grid size = 64×64, vertical capacity = 12, horizontal capacity = 14, and number of net = 11,507 nets) using routing cost based on a global routing congestion map. To evaluate the performance of the CAST, we perform experiments on a set of random nets and compare the routing costs of our CAST, CMST, and ER routing costs. We consider various cases started from 100 to 10,000 nets. These results are shown in Table 1. To our knowledge, there is no other publicly available Steiner tree heuristic algorithm which operates on a global routing graph. Detailed information of the costs is as follows.

• The "ER" or Edge Removal heuristic algorithm [25] cost columns indicate routing congestion cost, relative to a minimum spanning tree constructed with tree length as the objective. The length metric is a traditional comparison; our implementations of the edge-removal heuristic algorithm are comparable to those previously published. Because the Steiner tree is constructed regardless of the routing congestion metric, not all improvements performed by this heuristic algorithm are actually beneficial. As a result, overall solution quality may be degraded.

• The "CMST" or Capacitated Minimum Spanning Tree algorithm [25] cost columns show results when the minimum spanning tree constructed with rout-

Table 1: *Cost comparisons*

n	CAST	ER	CAST<ER	CMST	CAST<CMST
100	3.253	17.252	18.858%	6.830	47.636%
110	3.894	20.126	19.347%	7.826	49.756%
120	4.464	22.535	19.810%	8.642	51.657%
130	6.164	30.457	20.239%	11.553	53.355%
140	6.846	33.175	20.634%	12.476	54.871%
150	7.699	36.668	20.996%	13.692	56.223%
160	9.032	42.351	21.327%	15.723	57.446%
170	11.507	53.200	21.630%	19.656	58.543%
180	12.083	55.154	21.908%	20.296	59.534%
190	14.582	65.792	22.163%	24.128	60.433%
200	16.862	75.283	22.398%	27.529	61.252%
210	17.042	75.360	22.615%	27.488	61.999%
220	18.553	81.321	22.815%	29.598	62.895%
230	19.546	84.980	23.000%	30.871	63.314%
240	24.723	106.689	23.173%	38.693	63.895%
250	24.044	103.046	23.333%	37.317	64.432%
260	29.357	125.013	23.483%	45.213	64.930%
270	27.878	118.013	23.623%	42.632	65.393%
280	28.176	118.614	23.754%	42.804	65.824%
290	36.167	151.470	23.877%	54.611	66.227%
300	38.062	158.637	23.993%	57.147	66.603%
400	67.847	272.970	24.855%	97.823	69.357%
500	117.371	462.245	25.391%	165.254	71.024%
600	153.896	597.504	25.756%	213.332	72.139%
700	240.300	923.496	26.021%	329.466	72.963%
800	291.254	1110.776	26.221%	396.079	73.534%
900	406.294	1540.309	26.377%	549.050	73.999%
1000	447.142	1687.113	26.503%	601.229	74.371%
2000	2062.956	7618.676	27.078%	2712.856	76.044%
3000	4974.292	18239.863	27.272%	6493.859	76.600%
4000	8875.551	32429.245	27.369%	11545.013	76.878%
5000	13604.621	49601.989	27.428%	17658.159	77.044%
6000	20845.996	75895.532	27.467%	27018.191	77.155%
7000	25221.856	91733.606	27.495%	32656.125	77.235%
8000	33437.809	121522.775	27.516%	43260.475	77.294%
9000	41737.235	151595.284	27.532%	53965.659	77.340%
10000	58023.214	210648.089	27.545%	74987.349	77.377%

ing congestion cost as the metric, rather than interconnect length. The length column compares tree lengths; not surprisingly, trees constructed when considering congestion have slightly higher length. The cost of spanning trees obtained when considering congestion are slightly lower than when the congestion is ignored.

• The "CAST" columns show results from our heuristic Congestion Aware Steiner Tree approach. Using spanning trees constructed with congestion as an objective function, and then applying the edge removal heuristic algorithm, we obtain the best results for the cost metric.

We observe that when the congestion costs are not considered, tree lengths are slightly lower, but the cost of the trees is higher. The inverse is also true. Our approach clearly trades interconnect/wiring length for congestion reduction, as would be expected. The quadratic routing cost function can be shown as follows.

$$Cost = 86.2906 - (0.2559n) + (0.00056n^2) \qquad (1)$$

We set parameters for optimization with Simulated Annealing algorithm, and evaluate the CAST result with the congestion cost. Exponential cooling begins at initial temperature, $T_0 = 100$, and decreases in steps according to $T_{i+1} = \alpha T_i$ where $0 < \alpha < 1$. A high temperature $T\_high \geq 80$, the temperature decreases with $T_{i+1} = T_i - exp(T_0)/n$, where $n =$ number of net. Otherwise, at low temperature where $T\_low < 80$, it decreases with $T_{i+1} = T_i - 0.001$. The algorithm terminates when there is no improvement in the solution's fitness value in a few iterations or other termination criterion has been met.

Run times for our heuristic algorithm are low. Overall, our approach is $O(V^2)$, making it nearly as fast as the best spanning tree algorithms. As a result, our approach is suitable for large circuit designs.

In Table 1, we compare the routing cost of our CAST, ER and CMST approaches). The numbers of nets (n) are varying from 100 to 10,000. We compare our CAST algorithm with the capacitated minimum spanning tree (CMST) and the ER algorithm finding that our approach offers impressive reduction in congestion cost in average about 23.1% and 63.4%, respectively. When routing congestion is considered in spanning tree construction, the overall wiring length is slightly higher, but the tree cost is slightly lower. The best result (in terms of interconnect tree cost) are obtained by our Congestion Aware Steiner Trees (CAST). The routing results from our CAST algorithm are graphically illustrated in Fig. 7(a) for the number of nets ranging from 0 up to 1,000, and Fig 7(b) for a high number of nets ranging overall from 0 up to 10,000.

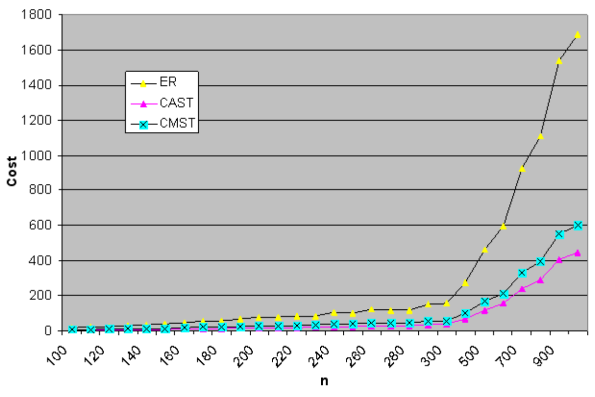

Fig. 7(a): *CAST, ER, and CMST routing costs for small to moderate net size*

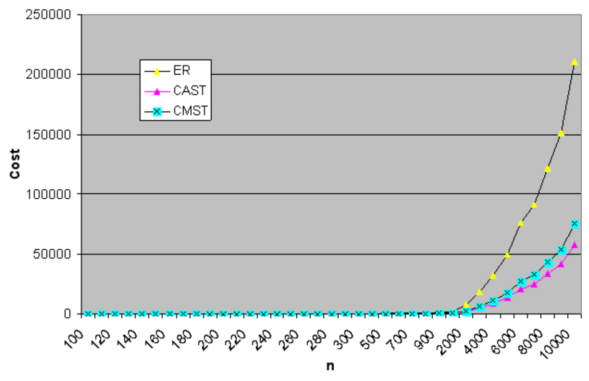

Fig. 7(b): *CAST, ER, and CMST routing costs for moderate to large net size*

In this paper, The CMST technique combines the speed and solution quality of a high quality Steiner heuristic algorithm with the reality of modern routing. Each routing layer has a preferred direction, and connecting vias have significant cost. We do not discuss obstacles and blockages; which are trivial to add to our formulation (by simply maintaining a count of the number of blocked routing tiles along with the summations). Our method also supports routing costs on a per-layer basis. For routing problems that contain large numbers of obstacles, maze-routing approaches would be more appropriate.

4. CONCLUSIONS

Routing is a key stage for VLSI physical design. This aggressive technology scaling has led to smaller/faster devices, but more resistive interconnects and larger coupling capacitance. Since routing directly determines interconnects (wiring length, routability, congestion, etc). The manufacturability and variability issues such as antenna effect, copper chemical mechanical polishing (CMP) and subwavelength printability have also emerged. Again, routing plays a major role in terms of the manufacturing closure. The global routing, as its name implies, is the routing stage that plans the approximate routing path of each net to reduce the complexity of routing task and guide the detailed routing. Thus, it significantly impacts on wiring length, routability and delayed time.

In this paper, we have presented the efficient Congestion Aware Steiner Tree (CAST) heuristic algorithm. From our observation that most global routing connections are simple L-shaped routes, we can use cached routing costs to speed up the calculations. Overall, our method is as fast as the best Steiner algorithms, which performs well when facing with a simple distance metric, and shifts to handle routing congestion costs easily. It also supports preferred direction routing, and the corresponding via costs.

As part of our current work, we are integrating the approach with our global routing tools [18]. We will report results of this integration when the work is completed.

5. ACKNOWLEDGEMENT

This work was supported by Grant for Development of New Faculty Staff Awarded 2001, Rajamangala University of Technology Isarn, Thailand. We also would like to thank Computer Science Dept. at Thomas J. Watson School of Engineering, State University of New York at Binghamton for supporting research facility during the year of 2006-2007.

References

[1] J. Cong, L. He, C.K. Koh, and P.H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal 21*, pp.1-94, 1996.

[2] M.C. Yildiz, and P.H. Madden, "Preferred direction multi-layer Steiner trees," *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems 21*, pp.1368-1372, Nov. 2002.

[3] M. Borah, R.M. Owens, and M.J. Irwin, "An edge-based heuristic for Steiner routing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 13*, pp. 1563-1568, Dec. 1994.

[4] C.Chiang, C.K. Wong, and M. Sarrafzadeh, "A weighted Steiner tree-based global router with simultaneous length and density minimization," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Trans*, Vol. 13, Issue 12, pp.1461 - 1469, Dec 1994.

[5] R.C. Prim, "Shortest connecting networks," *Bell System Technical Journal 31*, pp.1398-1401, 1957.

[6] J.B. Kruskal, "On the shortest spanning subtree of a graph," *Proc. American Math Society 7*, pp.48-50, 1956.

[7] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM Journal of Applied Mathematics 14*, pp.255-265, 1966.

[8] A.B. Kahang, and G. Robins, "A new class of iterative Steiner tree heuristics with good performance," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 11*, pp. 893-902, Jul. 1992.

[9] D.M. Warme, P. Winter, and M. Zachariasen, "Exact solutions to large-scale plane Steiner tree problems," *In Proc. 10th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA-99).* pp.979-980, 1999.

[10] M.M Ozdal, and M.D.F. Wong, "A history-driven global routing algorithm," *In Proc. Of IEEE/ACM Int. Conf. on Computer-aided Design*, California, pp. 488-495, 2007.

[11] C. Chu, "FLUTE: fast lookup table based wirelength estimation technique," *In Proc. Int. Conf. on Computer Aided Design*, pp. 696-701, 2004.

[12] J. Lou, S. Krishanmoorthy, and H.S. Sheng, "Estimating routing congestion using probabilistic analysis," *In Proc. Int. Symp. on Physical Design*, pp.112-117, 2001.

[13] R. Linsker, "An iterative-improvement penalty-function-driven wire routing system," *IBM journal of research and development 28*, pp.613-624, Sept. 1984.

[14] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik 1*, pp.269-271, 1959.

[15] M. Minoux, "Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity," *INFORMS Journal of Computing 28*, pp.221-233, Aug. 1990.

[16] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic congestion prediction," *In Proc. Int. Symp. on Physical Design*, pp.204-209. 2004.

[17] A. Agnihotri, and P.H. Madden, "Congestion

reduction in traditional and new routing architectures," *In Proc. Great Lakes Symposium on VLSI*, pp.211-214, 2003.

[18] R.T. Hadsell, and P.H. Madden, "Improved global routing through congestion estimation," *In Proc. Design Automation Conf*, pp. 28-31, 2003.

[19] M.Cho, and D. Z. Pan, "Box router: a new global router based on box expansion and progressive ILP," *In Proc. Design Automation Conf*, pp 24-28, Jul. 2006.

[20] X. Yang, M. Wang, R. Kastner, S. Ghiasi and M. Sarrafzadeh, "Congestion Reduction during Placement with Provably Good Approximation Bound," *ACM Transactions on Design Automation of Electronic Systems*, July 2003.

[21] J. Cong, and P. H. Madden, "Performance driven multi-layer general area routing for PCB/MCM designs," *In Proc. Design Automation Conf*, pp. 356-361, 1998.

[22] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "An extract algorithm for coupling-free routing," *In Proc. Int Symp. On Physical Design*, pp. 10-15, 2001.

[23] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 20(5)*, pp622-631. May 2001.

[24] A.R Agnihotri ,and P. H. Madden, "Fast analytic placement using minimum cost flow," *In Proc. Design Automation Conf ASP-DAC*, pp.128 - 134, Jan. 2007.

[25] J. Westra, P. Groeneveld, T. Yan, and P.H. Madden, "Global routing: Matrices, benchmarks, and tools.," *In IEEE DATC Electronics Design Process*, Apr. 2005.

[26] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 21*, Jul. 2002.

[27] R.C. Prim, "Shortest connecting network," *Bell System Technical Journal 31*, pp. 1398-1401, 1957.

[28] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, 1999.

Apichat Terapasirdsin received the BS degree in Electronics Engineering. (Second Class Honors) from King Mongkut's University of Technology Thonburi (KMUTT), Thailand in 1997. He received M.Eng. degree in Computer Engineering from King Mongkut's University of Technology Thonburi (KMUTT), Thailand in 2000, and he is currently pursuing a Ph.D. degree in Computer Engineering from KMUTT. During 2006-2007, he was the affiliated student at Department of Computer Science, State University of New York at Binghamton (SUNY), USA. His research interests are optimization for NP-hard problems (with applications to VLSI integrated circuit design) and fundamental problems in the VLSI CAD field, with an emphasis on high performance interconnect design, system level performance optimization, and novel approaches to routing and placement.

William F. Rotach is a B.S. student of Computer Sciences at the State University of New York at Binghamton. His research interest are high performance interconnect design, system level performance optimization, and novel approaches to routing and placement.

Naruemon Wattanapongsakorn received the B.S. degree in Computer Engineering in 1994, and the M.S. degree in Electrical Engineering in 1995, both from The George Washington University, and Ph.D. degree in Electrical Engineering in 2000 from the University of Pittsburgh. Her research interests include dependability analysis, optimization algorithms, embedded system modeling, and software fault-tolerance. She is currently an Associate Professor in Computer Engineering at the King Mongkut's University of Technology Thonburi (KMUTT), Thailand.

Patrick H. Madden received the B.S. degree in Computer Science and Mathematics, from New Mexico Institute of Mining and Technology in 1987, and the Ph.D. degree in Computer Science from University of California at Los Angeles in 1998. His research interests include fast placement for large circuits, novel VLSI routing architectures, and new Steiner tree formulations, optimization for NP-hard problems (with applications to VLSI integrated circuit design), high performance interconnect design, system level performance optimization, and novel approaches to routing and placement. He is currently an Associate Professor in Computer Science at State University of New York at Binghamton, New York, USA.